Twitter, Sensors and UI: Robust Context Modeling for Interruption Management

Justin Tang and Donald J. Patterson

University Of California, Irvine, USA
justinwktang@yahoo.com,
djp3@ics.uci.edu

Abstract. In this paper, we present the results of a two-month field study of fifteen people using a software tool designed to model changes in a user's availability. The software uses status update messages, as well as sensors, to detect changes in context. When changes are identified using the Kullback-Leibler Divergence metric, users are prompted to broadcast their current context to their social networks. The user interface method by which the alert is delivered is evaluated in order to minimize the impact on the user's workflow. By carefully coupling both algorithms and user interfaces, interruptions made by the software tool can be made valuable to the user.

1 Introduction and Related Work

Working with online tools makes vast networks of information and social resources available, but reciprocally exposes users to the reach of both software and people. Interruptions inevitably follow and have become a hallmark of modern information work while in the office [1, 2], while working nomadically [3], and while mobile [4].

"Interruptions", while generally perceived as negative, play an important role in the accomplishment of work and don't always have a correspondingly negative impact. Depending on their context, interruptions can be helpful or harmful and interruptees often express ambivalence toward their disruption and value [5]. An interruption from an assistant, for example, while disruptive at the moment, may provide critical information that will prevent wasted effort, enable effective response to time critical situations and reduce the need for future interruptions.

Interruptions are disruptive because they act in such a way as to break one's concentrated creative energy and give rise to feelings of loss of control [6]. Examples include receiving a phone call, having a supervisor enter a work space or having an instant message window appear. However, another class of interruptions is internally generated. Examples of these include suddenly remembering a forgotten task, a self-recognition of fatigue, or an unexpected insight into a forgotten problem. Internal interruptions are easier to accommodate fluidly however, because unlike a supervisor standing in front of you, the degree of response is under the control of the interruptee.

Recognizing technology's role in *creating* opportunities for interruption, many researchers have undertaken studies that attempt to evaluate technology's capability to help *manage* them as well. Initially such studies were directed at busy office managers and explored the potential for exposing interruptibility cues to colleagues, for example based on sensors in the office [7, 8] or changes in workflow on the desktop [9, 10]. As cell phones have grown in capabilities and prominence, researchers have expanded this sensor-based approach to address mobile individuals as well. Much of this work has been done to advance the methodological concerns associated with the experience sampling method, which invokes automated interruptions in user experience studies [11, 12], but has broad applicability in other domains.

This paper builds on these sensor-based approaches for determining interruptibility to support mobile laptop users. Our goal is to create software that will interrupt the user, not for conducting a user study, but for reminding the user to report their current context to their social network (for example through the status update service Twitter[1]). We hypothesize that such software can be effective because it has access to local sensor information, and will provide value to users by reminding them to inform their social network of changes to their availability before they are blindly interrupted from afar.

In this paper we report on a field study in which we compare the effectiveness of various approaches to accomplish this goal. We make three novel contributions. First, we leverage the user's current Twitter status as a *virtual sensor* in conjunction with existing sensors as input to a context modelling algorithm. Second we use a novel algorithm based on *Kullback-Leibler Divergence* (KLD) to detect changes in context. Third, we study the response of users to a variety of user interface (UI) techniques in order to identify properties of UI interruptions that make them able to be handled like an internal interruption by a user.

2 Background

Researchers have studied how instant messaging (IM) awareness cues have been appropriated by users as a means for inviting and discouraging interactions [13]. As with many social networking systems, IM users maintain buddy-lists in which they can report a custom status message that is broadcast to all of their contacts. The UIs of IM clients are designed in such a way that status messages can be viewed just prior to interrupting a remote user with a message. Users leverage this flexibility to, among other things, express their mood, promote issues and causes and *describe their current context* [14, 15]. Many of the practices around custom IM status messages have also been adopted in social networking systems such as Twitter and Facebook[2].

In previous studies we hypothesized that this last type of status message could be accurately guessed by modeling users' past behavior in light of sensor readings from the environment. Because of the complexity and privacy concerns related

[1] `http://www.twitter.com`
[2] `http://www.facebook.com`

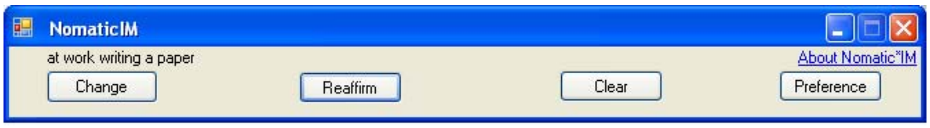


Fig. 1. Nomatic*IM steady state

to accurately representing a user [16] we avoid automatically broadcasting status from a predefined ontology, opting instead to make it easy for a user to manually update their status in their own words. To test this hypothesis we built a status message broadcast tool called Nomatic*IM [4]. This tool gives users a single point of entry for updating their status across a wide variety of IM and web services. In previous studies we observed that this single-point-of-entry strategy was effective and helpful in assisting users in managing social contact [13]. However, effectiveness of, and satisfaction with the tool were closely correlated to how frequently the status messages were updated.

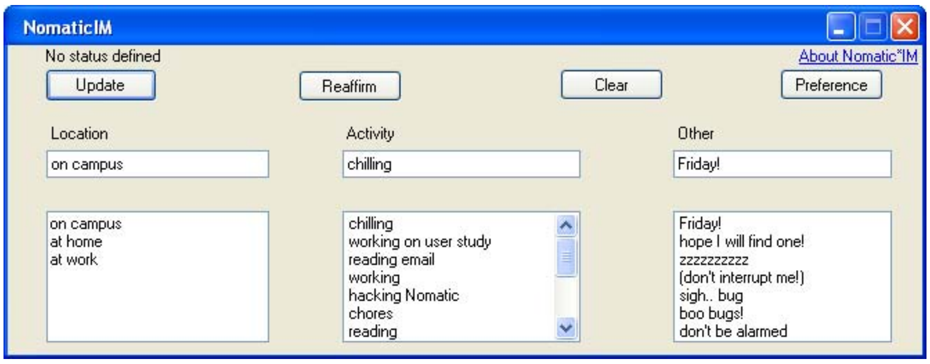
Previously, to keep status updates accurate, simple hand-crafted rules were used to recognize changes in a user's context [17]. Rules based on elapsed time were the most effective, but contrary to our hypothesis, rules based on sensors proved to be ineffective because of the complexity of the sensed environment. For example, although a change in IP address might be assumed to be associated with a change in location, and subsequently a change in availability, in practice, IP addresses change for a wide variety of reasons not associated with mobility. Temporary network disconnections, automatic switching between co-located WiFi access points, and DNS credential refreshing all caused fluctuations in IP addresses. This motivated the more robust treatment of sensor data described in this paper.

3 User Interface

Because of the many ways in which people use status updates that are not related to availability context, we developed the user interface of Nomatic*IM to guide the user in making the type of status updates that are relevant for our study.

The steady state Nomatic*IM interface is shown in Fig. 1. In this window the user's current status is displayed while the software collects readings from sensors in the background. There are buttons that allow a user to initiate a change in status, rebroadcast the current status (reaffirm), clear the current status, or manage their preferences.

When users want to change their status, the window is extended to present three fields in which a sentence can be constructed consisting of a location, an activity, and a free form field. The user can type in new data or use historic data that is presented in drop down lists as shown in Fig. 2. When the user is satisfied and accepts the current status by clicking the update button, Nomatic*IM makes a current reading of the sensors, pairs them with the current status message, stores them locally in order to later form a user model and then broadcasts the status message out to user-configured services such as Skype, AIM, Twitter, Facebook, etc.


NomaticIM

No status defined

About Nomatic*IM

Update | Reaffirm | Clear | Preference

Location	Activity	Other
on campus	chilling	Friday!
on campus at home at work	chilling working on user study reading email working hacking Nomatic chores reading	Friday! hope I will find one! zzzzzzzzzz (don't interrupt me!) sigh.. bug boo bugs! don't be alarmed

Fig. 2. Nomatic*IM status entry dialog

In addition to UI elements for the user-initiated flow of events, we also implemented and subsequently studied the acceptance of five interruption techniques to alert the user when the system identified that a context change had occurred.

1. PopUp-Window-UI, which brings the window from Fig. 2 to the forefront
2. Cursor-Change-UI, which changes the mouse pointer to a custom icon until the user updates their status (see Fig. 3 left).
3. Fading-Window-UI, which creates a popup window of varying opacity depending on the urgency of the interruption (see Fig. 3 center).
4. Systray-Balloon-UI, which activates a small alert in the bottom right system tray (see Fig. 3 right).
5. Audio-Interrupt-UI, which plays a short audio chime.

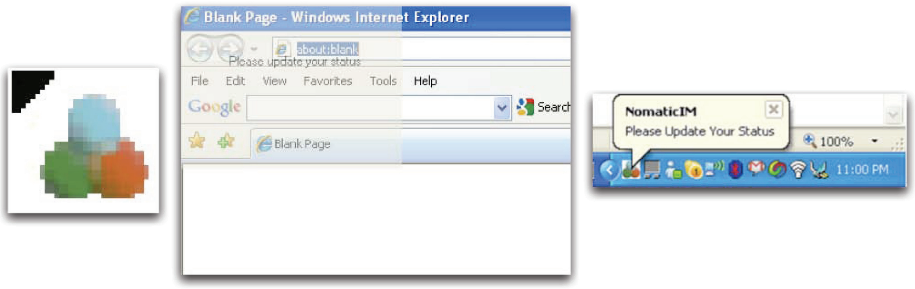

Blank Page - Windows Internet Explorer
about:blank
Please update your status
File Edit View Favorites Tools Help
Google
Search
Blank Page
NomaticIM
Please Update Your Status
100%
11:00 PM

Fig. 3. Cursor-Change-UI (left), Fading-Window-UI overlaid on a web browser window (center), Systray-Balloon-UI (right)

4 Robust Context Change Detection

To robustly detect a change in context, we developed an alternative to the previously studied rule-based strategy. It is based on the the Kullback-Leibler Divergence (KLD), a non-negative numeric measure of how similar two multinomial

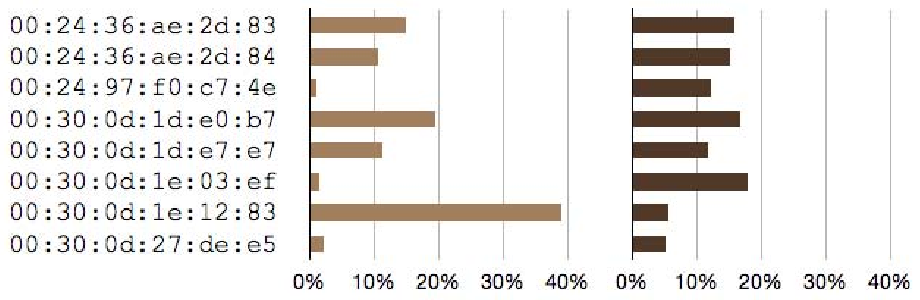


Fig. 4. Example probability distributions for the WiFi MAC address sensor when at the place, "at home", $X^{WiFi_{\text{“}at\_home\text{”}}}$ and $Y^{WiFi_{\text{“}at\_home\text{”}}}$

probability distributions are. A value of 0 indicates exact similarity. Formally, KLD measures the number of extra bits required to encode a distribution, Y, using the information in another distribution, X. Informally, KLD can be used to measure the similarity between two normalized histograms. It is defined as follows:

$$D_{\text{KL}}(X\|Y) = \sum_i X(i) \log_2 \frac{X(i)}{Y(i)}$$

While using Nomatic*IM, our participants built up a history of contextual status messages and sensor readings. The sensor readings were collected both when the context message was changed, but also periodically in the background. During this study, sensor readings were collected every 60 seconds. A wide variety of sensors were captured including current time, network parameters, battery status, display configuration, WiFi parameters, UI activity and active processes. Depending on the hardware available additional sensors such as light and sound levels, accelerometer readings, and location were collected as well. With the exception of the status message, no semantic content (e.g., keystrokes, IM content, browser URLs) was collected.

In order to detect a change in context, a collection of probability distributions were constructed using the historic sensor readings that had been observed by the system when the user was reporting their status. Separate distributions were calculated for each of the three status field components. When users changed their status message, $X^{S_{place}}, X^{S_{activity}}, X^{S_{other}}$ were selected from the collection of historic probability distributions based on the three status fields. $X^{S_{place}}$ subsequently contained the probability $P(S = i|place = p)$, where i ranges over the possible values of the sensors and p is the current value of "place" (or "activity" or "other"). A second set of three distributions were created and augmented every time a new sensor reading was taken in the background. These became the distributions, $Y^{S_{place}}, Y^{S_{activity}}, Y^{S_{other}}$. Each of the three distributions differ from the others because they use the subsets of sensors that are relevant for modeling place, activity and other.

The change detection algorithm is based on a comparison of the historic distribution of sensors, X, compared with the currently observed sensors, Y, given the most recently entered status message. The comparison is captured in the values $D_{\mathrm{KL}}(X^{S_{place}} \| Y^{S_{place}})$, $D_{\mathrm{KL}}(X^{S_{activity}} \| Y^{S_{activity}})$, $D_{\mathrm{KL}}(X^{S_{other}} \| Y^{S_{other}})$.

Figure 4 shows an example of these distributions. The probability distribution on the left shows an example of the distribution of WiFi MAC addresses that have ever been historically collected for a user when reporting being "at home" in the place field of their status. This is the distribution, $X^{S_{place}}$. The figure on the right is the distribution over the currently observed WiFi MAC addresses since the last update when the user entered "at home" as their current location. This is $Y^{S_{place}}$. Our algorithm monitors $D_{\mathrm{KL}}(X \| Y)$ and after collecting five minutes worth of samples, alerts when the metric begins to rise. An increasing value is associated with diverging distributions between the sensor values historically expected and currently observed.

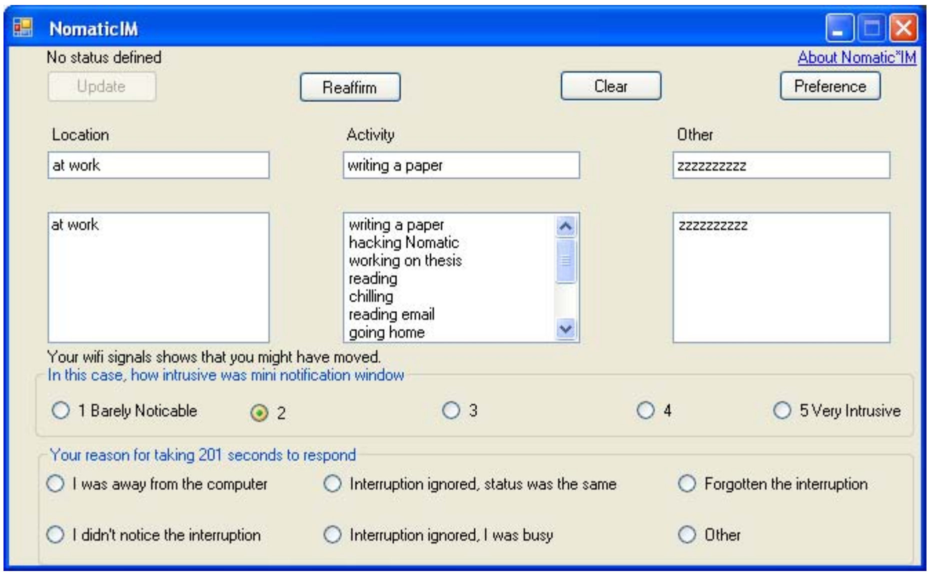
5 Methodology

We conducted a within-subjects experiment across two variables to compare the techniques that we developed. The first variable was the UI element that was used to alert the user. It took on one of the five conditions corresponding to the techniques specified in Sect. 3.

The second variable was the context change detection algorithm. This variable had six conditions corresponding to five rules plus our new KLD technique.

1. WiFi-MAC-Change-R: This rule asserted a context change when a user's laptop connected to a new WiFi access point as determined by the MAC address.
2. Local-IP-Change-R: This rule asserted a context change if the local network IP address changed on the primary adaptor.
3. Remote-IP-Change-R: This rule detected if the IP address of the user's laptop changed as viewed from the internet (the "remote" IP address).
4. Stale-R: This rule was activated if the user did not change their status for 2 hours.
5. Start-Up-R: This rule activated if the user did not change their status within 3 minutes after starting the software.
6. KLD-R: This technique detected a change the first time that a user's current distribution of WiFi access point MAC addresses diverged from the historic distribution for the same place status or when the current distribution of active processes diverged from the historic distribution of active processes for the current activity status. All techniques were prevented from activating within five minutes of the last change.

Each time the user entered a new status or focussed on our tool following an interruption, we randomly selected a new pair of conditions to test. When the Fading-Window-UI was selected with the KLD-R alert technique, the window's opacity was set based on the KLD measure such that as the two distributions diverged more, the window became more opaque. When a rule based alert was paired with Fading-Window-UI the opacity was set to 75%.


NomaticIM

No status defined

About Nomatic*IM

Update | Reaffirm | Clear | Preference

Location	Activity	Other
at work	writing a paper	zzzzzzzzzz
at work	writing a paper hacking Nomatic working on thesis reading chilling reading email going home	zzzzzzzzzz

Your wifi signals shows that you might have moved.

In this case, how intrusive was mini notification window

1 Barely Noticable [✓ 2] 3 4 5 Very Intrusive

Your reason for taking 201 seconds to respond

- I was away from the computer
- Interruption ignored, status was the same
- Forgotten the interruption
- I didn't notice the interruption
- Interruption ignored, I was busy
- Other

Fig. 5. UI following a long delay

Additionally when Nomatic*IM initiated an alert, the status change window was augmented with a description of why the system initiated an alert and a question asking the user to rate how "intrusive" this alert was on a 5-point Likert scale. To ensure that the data that we collected didn't have cofounding temporal factors, we monitored the time for the user to respond to an alert. If the user took more than three minutes to respond to the interruption, the user was asked a second question about why it took so long to respond, the choices included: "I was away from the computer", "I didn't notice the interruption", "Interruption ignored, my status was the same","Interruption ignored, I was busy", and "Other" which allowed for a free-form response (see Fig. 5).

Fifteen participants were recruited from the University of California, Irvine community via email advertisement and mailing lists, website advertisements, and flyers. Participants were also recruited by in-person invitations. We enrolled adults who use Windows XP or Windows Vista laptops as well as Skype, Facebook, and/or Twitter, and who self-reported using their laptop in more than 2 physical locations per day via WiFi. Upon enrollment, participants were provided with a copy of Nomatic*IM, and given step-by-step instructions via an online video on installation and usage. Participants were instructed to use their computer normally, with the exception of setting their status via Nomatic*IM. Participants were compensated $1.00 for each day that they entered a status up to a maximum of $42.00. The study lasted a total of two-months.

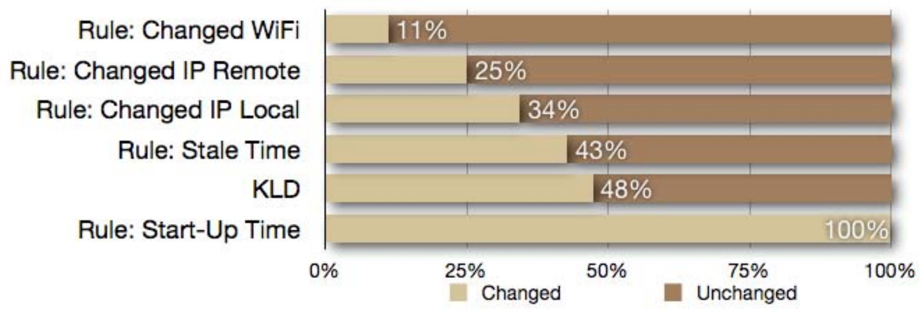


Fig. 6. The effectiveness of various context change detection techniques. 100% changed is the desired outcome.

6 Data Analysis

During the course of our study, our participants made a total of 1157 changes to their status. Although there was a great deal of individual variance in users, some broad trends could be observed. We were interested in looking at the relative effectiveness of the context change detection algorithms, how different UI techniques affected status updating behavior and overall user satisfaction with the techniques.

Status Change. Figure 6 compares different ways of detecting context change and whether or not users ultimately changed their status in response to the alert. A change in status is our ground truth. We are not interested in knowing whether the user's status is "true" in some understandable way (as is typically the case in the activity recognition literature), but rather whether our system successfully supported the user in keeping their status up to date. We would like a technique that only alerts users when they want to change their status. An algorithm that is too aggressive in suggesting that a user's context has changed, requiring a status update, risks disrupting the user too often. An algorithm that is too conservative fails to provide value.

Unsurprisingly, START-UP-R was 100% effective in prompting a status change. This rule coincides with a user starting the program which is likely prompted by a self-initiated need to broadcast a status or the initial boot process of the laptop. The least effective context detection algorithms were sensor-based rules. WIFI-MAC-CHANGE-R was only successful 11% of the time, followed by rules detecting changes in IP address. REMOTE-IP-CHANGE-R was accurate 25% of the time and LOCAL-IP-CHANGE-R was accurate 34% of the time. Simply suggesting a change after a period of time had elapsed was effective 43% of the time, which likely captured the scenario when a user awakened their laptop from a sleep state. KLD-R managed to outperform all the non-trivial rules with a 48% success rate and addresses the situations meant to be captured by the 3 least successful rules. All of the results in Fig. 6 showed a statistically significant difference from KLD-R ($p < 0.05$).

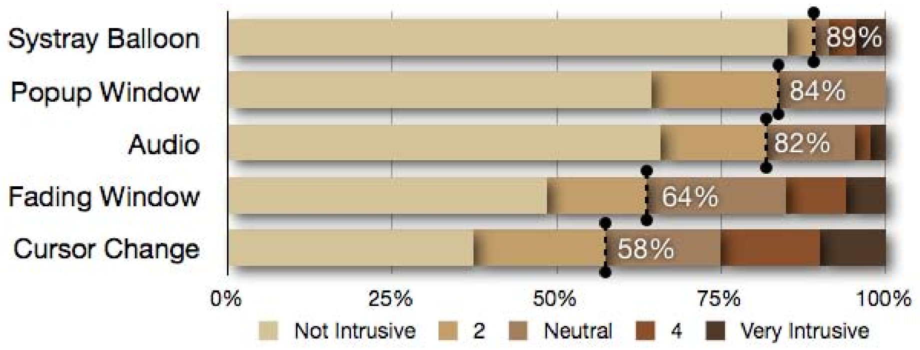


Fig. 7. User reported intrusiveness levels of various UI interruption methods

Intrusiveness Level. An aggregate analysis of user evaluation of different user interface interruption techniques is visualized in Fig. 7. This figure lists each of the 5 different techniques that were used to get the attention of the user when one of the context change detection algorithms alerted. The percentage of Likert scale ratings is shown as horizontal bars totalling 100%.

Treating a rating of 1 or 2 (low intrusiveness) as being acceptable, our data shows that the SYSTRAY-BALLOON-UI technique was acceptable 89% of the time, followed by the POPUP-WINDOW-UI 84% of the time, the AUDIO-INTERRUPT-UI 82% of the time, the FADING-WINDOW-UI 64% of the time and the CURSOR-CHANGE-UI 58% of the time. The difference in preference between the AUDIO-INTERRUPT-UI and the FADING-WINDOW-UI techniques was statistically significant ($p < 0.05$).

Long Response. During the study we evaluated the length of time it took before our participants would respond to the UI alerts. Our goal was to attempt to appropriately credit changes to status that were caused by our alerts, as opposed to changes in status that were made by a user who did not see an alert and later self-initiated. The reasons for the long alerts are shown in Fig. 8. Across the board, the most common reason for the prolonged response is that the users did not notice the interruption. This suggests that mobile users may not be as focussed on their laptops as we had assumed. This point is particularly highlighted by the responses to POPUP-WINDOW-UI. 55% of the responses were that the user did not notice the interruption. Considering that this particular interruption would take up a majority of the real estate on screen, and tying this data with the fact that POPUP-WINDOW-UI is scored as *not intrusive* 68% of the time, suggests that our participants were not completely focused on their laptops while they were running.

When a user did notice an alert, the decision to change their status was not statistically significantly correlated with the UI technique used to alert the user. This is a reasonable outcome and suggests that no UI technique caused the user to want to broadcast a different status just by its use. We were able to track a no-change-but-noticed case because a user can notice an alert, switch focus

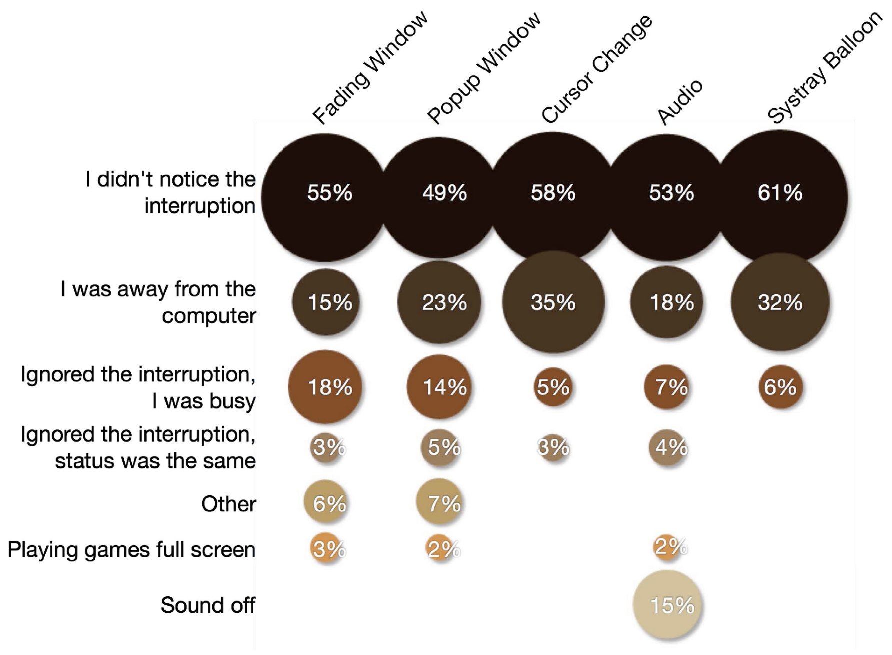


Fig. 8. Why participants took a long time to respond to UI alerts. Each column is normalized to reflect relative importance of each reason given a UI technique.

to our tool and explicitly tell our tool to rebroadcast the same status using a button for this purpose. This clears any persistent alerting mechanism, and our users did this on many occasions even for non-persistent alerts.

7 Conclusion

At the beginning of this study, we identified differences between internal and external interruptions and suggested that users would be more satisfied with external interruptions that could be handled more like an internal interruption. This was the motivation for creating the Fading-Window-UI and Cursor-Change-UI elements. They are both small and persistent but easily ignorable.

Empirically, however, we observed that Fading-Window-UI and Cursor-Change-UI were the most likely UI techniques to be rated as intrusive. One explanation for this result is that since these techniques were developed for this study, they were novel to our participants. Encountering these UI interruptions may have seemed more intrusive simply because they were unfamiliar experiences.

Although Cursor-Change-UI was scored as the most intrusive UI technique, it was one of the most *effective*, as only 8% of the time was the interruption method ignored. However, Systray-Balloon-UI also had a similar low probability of being ignored (6%), but was evaluated as being the least intrusive. One explanation for this difference could be that non-persistent interruption techniques can be just as effective as persistent techniques, while being much less irritating. More support for this position comes from the AUDIO-INTERRUPT-UI, which plays its chime only once and was rated as non-intrusive. These alerts may be more easily mentally shifted from being an external interruption to being an internal interruption and thus able to be attended to at a natural break in the user's workflow. A competing hypothesis is that non-persistent interruption methods may also be perceived as less intrusive because users are away from their computer, or the sound is off, when the interruption occurs. More evaluation is required to isolate this effect.

This user study of context modeling and interruption techniques demonstrates that using the Kullback-Leibler Divergence metric to model context with inputs of both traditional sensors and Twitter-like status messages as *virtual sensors* can be more effective than hand-coded rules based on traditional sensors alone. The data suggest that an optimized context change detection would use a hybrid of rules based on elapsed time coupled with robust evaluations of the sensed environment and user entered status updates.

Our motivation was to create a software application that broadcasts context for social interruption management. As such it was important that the interruptions done by our software were as non-intrusive as possible, so that, like a good human assistant, the user sees interruptions by the software as an overall benefit. We demonstrated that although context change can be modelled with some accuracy, that is not the complete picture. User satisfaction is also heavily influenced by the UI technique and timing that is used to present alerts to the user. Our data showed that effective techniques for getting users' attention are not necessarily the ones that they prefer. Our participants preferred non-persistent, familiar UI alerts that we hypothesize can be more easily internalized until a natural break in the workflow permits them to be attended to.

References

1. González, V.M., Mark, G.: Constant, constant, multi-tasking craziness: Managing multiple working spheres. In: Dykstra-Erickson, E., Tscheligi, M. (eds.) CHI, pp. 113–120. ACM, New York (2004)
2. Rouncefield, M., Hughes, J.A., Rodden, T., Viller, S.: Working with "constant interruption": CSCW and the small office. In: CSCW, pp. 275–286 (1994)
3. Su, N.M., Mark, G.: Designing for nomadic work. In: van der Schijff, J., Marsden, G. (eds.) Conference on Designing Interactive Systems, pp. 305–314. ACM, New York (2008)
4. Patterson, D.J., Ding, X., Noack, N.: Nomatic: Location by, for, and of crowds. In: Hazas, M., Krumm, J., Strang, T. (eds.) LoCA 2006. LNCS, vol. 3987, pp. 186–203. Springer, Heidelberg (2006)
5. Hudson, J.M., Christensen, J., Kellogg, W.A., Erickson, T.: "I'd be overwhelmed, but it's just one more thing to do": Availability and interruption in research management. In: CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 97–104. ACM, New York (2002)

6. Csikszentmihalyi, M.: Flow: The Psychology of Optimal Experience. Harper Perennial (March 1991)
7. Horvitz, E., Koch, P., Apacible, J.: Busybody: creating and fielding personalized models of the cost of interruption. In: Herbsleb, J.D., Olson, G.M. (eds.) CSCW, pp. 507–510. ACM, New York (2004)
8. Fogarty, J., Hudson, S.E., Atkeson, C.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., Yang, J.: Predicting human interruptibility with sensors. ACM Trans. Comput.-Hum. Interact. 12(1), 119–146 (2005)
9. Iqbal, S.T., Bailey, B.P.: Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. In: Rosson, M.B., Gilmore, D.J. (eds.) CHI, pp. 697–706. ACM, New York (2007)
10. Shen, J., Irvine, J., Bao, X., Goodman, M., Kolibaba, S., Tran, A., Carl, F., Kirschner, B., Stumpf, S., Dietterich, T.G.: Detecting and correcting user activity switches: algorithms and interfaces. In: Conati, C., Bauer, M., Oliver, N., Weld, D.S. (eds.) IUI, pp. 117–126. ACM, New York (2009)
11. Ho, J., Intille, S.S.: Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In: van der Veer, G.C., Gale, C. (eds.) CHI, pp. 909–918. ACM, New York (2005)
12. Kapoor, A., Horvitz, E.: Experience sampling for building predictive user models: a comparative study. In: CHI, April 2008, pp. 657–666 (2008)
13. Ding, X., Patterson, D.J.: Status on display: a field trial of Nomatic*Viz. In: Wagner, I., Tellioğlu, H., Balka, E., Simone, C., Ciolfi, L. (eds.) ECSCW 2009. Computer Science, pp. 303–322. Springer, London (2009)
14. Smale, S., Greenberg, S.: Broadcasting information via display names in instant messaging. In: GROUP '05: Proc. of the 2005 Intl ACM SIGGROUP Conference on Supporting group work, pp. 89–98. ACM, New York (2005)
15. Cheverst, K., Dix, A., Fitton, D., Rouncefield, M., Graham, C.: Exploring awareness related messaging through two situated-display-based systems. Hum.-Comput. Interact. 22(1), 173–220 (2007)
16. Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing 8(1), 19–30 (2004)
17. Patterson, D.J., Ding, X., Kaufman, S.J., Liu, K., Zaldivar, A.: An ecosystem for learning and using sensor-driven IM messages. IEEE Pervasive Computing 8(4), 42–49 (2009)